CS4320 Prelim Exam

March 21th, 2017
(50 minutes working time)

Name: _____ Cornell NETID: _____

I understand and will adhere to the Cornell Code of Academic Integrity.

Signature

Maximum number of points possible: 40. This exam counts for 20 % of your overall grade. Questions vary in difficulty. Do not get stuck on one question.

In all problems, whenever you think a problem is underspecified, make assumptions and clearly state them.

Good luck!

Note – you have 50 minutes working time for this exam, NOT 2 hours as on some other prelims.

Part A) SQL Queries. (10 points)

Consider the database schema created by the following SQL commands:

```
CREATE TABLE Products(productID INTEGER PRIMARY KEY,
productName VARCHAR(30));

CREATE TABLE Orders(orderID INTEGER PRIMARY KEY,
productID INTEGER,
FOREIGN KEY (productID) REFERENCES Products(productID));
```

This database contains information about products (table `Products`) and orders of those products (table `Orders`).

A.1) Write an SQL query retrieving the names of products that have not been ordered yet. (5 points)

```
SELECT P.productName, P.productID from Products P
WHERE NOT EXISTS (
     SELECT O.productID FROM Orders O
     WHERE O.productID = P.productID);
```

A.2) Write an SQL query retrieving the names of those products that have been ordered at least five times. (5 points)

```
SELECT P.productName, P.productID
FROM Products P, Orders O
WHERE P.productID = O.productID
GROUP BY P.productID
HAVING COUNT(*) >= 5;
```

Part B) External Sorting. (10 points)

In the lecture, we have seen an algorithm that sorts large data sets which do not fit into main memory. This algorithm performs multiple passes on the data and writes out intermediate results to hard disc during each pass. In this exercise, we ask you to execute that sorting algorithm by hand on an example data set and note down the results that are written to disc after each pass.

We have three pages of main memory available (which are initially empty). We assume that each page holds only a single number. The input data, stored on hard disc, looks as follows:

4	3	7	1	2	5	9	2	3	8	1	5

We have seen a naive algorithm in class and an algorithm that exploits main memory efficiently. We ask you to execute the efficient version in the following. Assume that input data is read from left to right and output is written from left to right as well. Fill in the content that is written to hard disc during each of the three passes.

Use **this page** to write your final solution and the **next** (blank) page for intermediate calculations.

Fill in the data written to hard disc after the first pass (pass 0) of the algorithm:

3	4	7	1	2	5	2	3	9	1	5	8

Fill in the data written to hard disc after the second pass of the algorithm:

1	2	3	4	5	7	1	2	3	5	8	9

Fill in the data written to hard disc after the third pass of the algorithm:

1	1	2	2	3	3	4	5	5	7	8	9

(This page is intentionally left blank)

Part C) Join Operators. (15 points)

We want to join two relations R and S using an equality join condition. Relation R contains 1000 tuples and consumes 100 disc pages. Relation S contains 2000 tuples and consumes 200 disc pages. We have 50 main memory pages available.

We ask you to calculate the execution cost (measured as the number of page I/Os) for different join operators. Assume that the values in the join column are unique within their respective relation (i.e., each value appears at most twice in the join column, once in R and once in S). Do not count the cost of writing out the final result but all other cost such as reading the input data.

C.1) Assume that we join R and S via a block nested loop join with R as outer relation. Assume that we use as many buffer pages as possible to read R (to maximize block sizes). (5 points)

We need one output buffer and one input buffer for S so 50 - 2 = 48 buffer pages remain for reading blocks of R. This means that we have to read three blocks from R. Three is therefore the number of iterations of the outer join loop and we read 200 pages (size of S) in each iteration. Taking into account the additional cost of reading R (100 pages), the total cost is
100 + 3 * 200 = 700 page I/Os.

C.2) Assume that we join R and S via a sort-merge join. Neither R nor S are initially ordered. Apply the most efficient variant of the sort-merge join that is applicable with the given amount of main memory. (5 points)

The refined version of the sort merge join is applicable since the amount of main memory exceeds the (double of) the square root of the number of disc pages in the larger relation (200).

Hence, the cost of the join is the cost of reading and writing both input relations during the first pass plus the cost of reading both input relations during the second pass:
3 * (100 + 200) = 900 page I/Os.

C.3) Assume that a hash index is available on S and on the join column. We assume that each index access (including the cost of retrieving the associated data) costs exactly two page I/Os. Calculate the cost of an index nested loop join that iterates over R as outer relation. (5 points)

The number of index accesses equals the number of tuples in the outer relation which is 1000. Hence, the total cost for index access is 1000 * 2 = 2000 page I/Os. In addition, we need to count the cost of reading R which is 100 page I/Os. In total, the cost of the index nested loop join is 2000 + 100 = 2100 page I/Os.

Part D) Index Selection. (5 points)

Propose one single index that can be used to answer **both** of the following queries from the index alone (i.e., without directly accessing the input relation `R`):

```
SELECT R.a, R.b, R.c FROM R WHERE R.a=10 AND R.b=20 AND R.c=10;
```

```
SELECT R.a, R.b FROM R WHERE R.a = 20 AND R.b < 10;
```

Describe the type of the proposed index (tree versus hash index), the columns to use as index key, and (if applicable) the ordering of those columns when creating the index. Justify with one or two sentences for each of the aforementioned three aspects why your choice is a good one.

We should create a tree index to support the inequality predicate in the second query. The index needs to contain columns R.a, R.b, and R.c to cover all data required by the two queries. The index key columns are ordered as (R.a, R.b, R.c) since a tree index can only match conditions that refer to columns forming a prefix of its index key order.

CS4320 Prelim Exam

This page will be used for grading your exam. Do not write anything on this page.

SECTION	QUESTION	SCORE	SECTION TOTAL
Part A	**A.1 (Max: 5 points)**		**(Max: 10 points)**
	A.2 (Max: 5 points)		
Part B	**B (Max: 10 points)**		**(Max: 10 points)**
Part C	**C.1 (Max: 5 points)**		**(Max: 15 points)**
	C.2 (Max: 5 points)		
	C.3 (Max: 5 points)		
Part D	**D (Max: 5 points)**		**(Max: 5 points)**
Total (Max: 40 points)			